Experiences with SICS

Mark Kőnnecke
Laboratory for Development and Methods
Paul Scherrer Institut
Mark.Koennecke@psi.ch

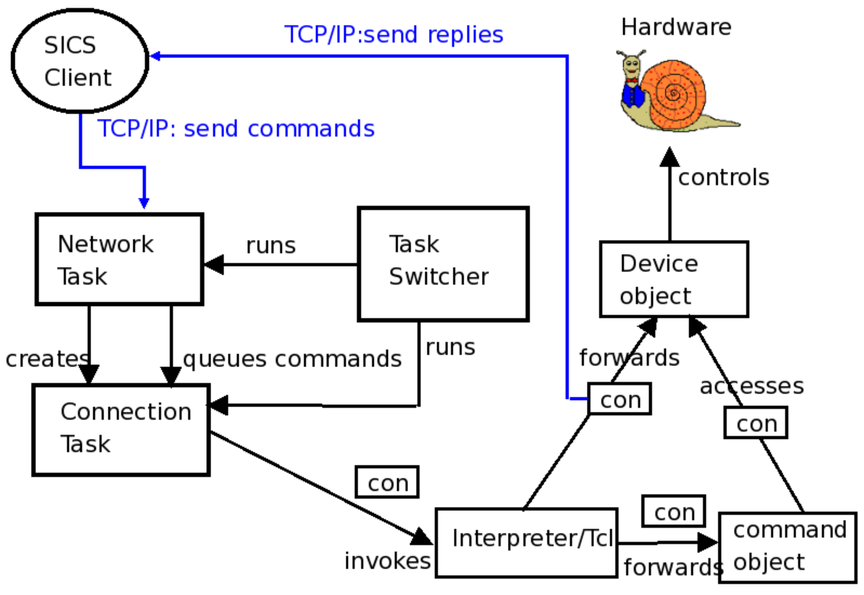
Introduction

Most instruments at the spallation neutron source SINQ use the SICS instrument control software. SICS is a client server system. The SICS server, written in ANSI-C and Tcl, does all the hard work and the clients implement the user interface. Currently several command line applications and various graphical status display clients and a full graphical user interface, gtse, all written in Java are used as Clients.

The SICS server has the following characteristics:
- TCP/IP Server with cooperative multitasking
- SICS server does all: HW access, control, NeXus data file writing, etc.
- OO-Design realised in ANSI-C with Tcl as serverside scripting language
- Single executable for 14 instruments, configured via configuration script
- Extendable command interpreter modelled after Tcl
- Role based access control
- Hierarchical parameter database
- Simple ASCII protocoll over sockets for client communication

This poster highlights the lessons which were learned from having SICS in operation since 14 years.

SICS Schema



What Did Not Work

- Hiding parameters from instrument scientists
- Role based access control, is routinely circumvented by logging in with highest privilege
---- It may be better to implement an ability to lock a parameter or a group of parameters
- Of 25 instrument scientists only a handful programs
- Do not expect IT knowledge from scientists
- Instrument scientists rarely know hat they need or want, maximum flexibility needed
- Synchronous I/O with devices leads to lockup
- Division between logical and device parameters
- Integration of commercial, closed source, software is always a problem
- Java garbage collection of arrays rarely works for reasons not understood

Lessons Learned

- Humans and Computers need different Protocols
-- Humans want descriptive text
-- Computers need clear termination and efficient bulk transfer of data
- 2/3 of the code is error processing, having working garbage collection and exception handling would help
- Efficient internal array processing is required
- Our instrument scientists love the command line, GUI's not well received

Problematic

- Observer pattern
---- Causes core dumps in C
---- Builds references in Java which may cause trouble in garbage collection
- Java graphics packages usually implemented as many small classes
---- Objects tend to be interconnected; this again causes problems in garbage collection when updated frequently
---- Better: plot data class plus utility classes to plot according to data
- Complex frameworks
---- Every framework is also a prison
---- Things not forseen in the framework can be difficult to express
---- Always weigh what the framework does for you against the dependency it brings

What Worked

- Serverside scripting language
- Scripted Configuration
- Common parameter model with: set, get, update, notify (and lock?)
- Context object to maintain client state
- Internal Interfaces for Drivables, Countables etc.
- Direct Controller Access
- HTTP Device Servers
- SW and HW standards allow serious understaffing
- Remote instrument access
- Widget command syntax
- Hierarchical DB to capture parameters, commands and online data
- Continuous refactoring
- Java indeed is well portable
- Deployment by Java webstart

Summary

The SICS concept of a single process data acquisition system proved to be an appropriate system for neutron scattering instruments with up to 60 motors and 4 detectors.

There was no guidance available when SICS was started in 1996. And there is none for newcomers in data acquistion roles now if they do not have the privilege to be able to join an existing group. Who writes a book on DAQ systems?